Modélisation et Conception Objet

TD 2 : Modélisation

Nous allons modéliser le jeu de carte de la bataille : [https://en.wikipedia.org/wiki/War_\(card_game\)](https://en.wikipedia.org/wiki/War_(card_game))

On distribue l'ensemble d'un (ou plusieurs) jeu de cartes (54 ou 32) aux joueurs, qui n'en prennent pas connaissance. À chaque tour, chaque joueur retourne la carte du haut de sa main (ou son tas). Celui qui a la carte de la plus haute valeur — selon la hiérarchie du bridge : As, Roi, Dame, Valet, dix... jusqu'au deux — fait la levée, qu'il place sous son tas.

En cas d'égalité de valeurs — cas appelé bataille — les joueurs en ballottage disent "bataille !", et commencent par placer une première carte face cachée puis une seconde carte face visible pour décider qui fera la levée. En cas de nouvelle égalité, la procédure est répétée. La bataille est parfois l'occasion d'acquérir une grosse carte et c'est l'unique manière de gagner un as. Sans bataille et à moins qu'un joueur ne possède tous les as, il serait impossible de terminer une partie de bataille. La partie se termine et le gagnant est déterminé lorsque ce dernier a en sa possession toutes les cartes du jeu.

1 Carte

1.1 Le modèle UML

Modélisez (sous format UML) une Carte :

- elle devra avoir une couleur et un nombre, donné à sa création
- une image associée. On y accède avec la méthode `image()`
- une représentation sous forme de chaîne de caractères.

1.2 Comparaison

Le jeu de la bataille nécessite des comparer des cartes. Proposez une méthode pour tester :

- l'égalité
- < et >

1.3 Code

Python permet de comparer des objets comme des entiers avec les méthodes :

- `__eq__` qui permet d'utiliser `==`
- `__lt__` qui permet d'utiliser `<` et `__gt__` qui permet d'utiliser `>`

Ecrivez la classe `Card`.

1.4 Mutable vs immutable

Doit-on pouvoir modifier une carte ? Quel est l'intérêt d'avoir une classe non mutable ?

1.5 Utilisation du code

Créez un jeu de 52 cartes que vous placerez dans une liste.

2 Deck

Chaque joueur joue à son tour une (ou plusieurs s'il y a bataille) carte de son `deck` (tas de cartes) :

- on doit pouvoir connaître le nombre de cartes du deck
- chaque carte est jouée du dessus du deck et les cartes gagnées sont placées en bas du deck
- un deck doit également pouvoir être mélangé en début de partie.
- on doit pouvoir ajouter toute les cartes d'un autre deck
- voir la première ou la dernière carte du deck

2.1 UML

Proposez une modélisation UML d'un Deck en précisant pour chaque méthode son fonctionnement.

2.2 Paquet de cartes

Donnez le pseudo-code d'une fonction `card_pack` qui rendra deux decks contenant chacun 26 cartes distribuées aléatoirement à partir d'un jeu de 52 cartes.

3 Jeu

Comment modéliser le jeu en lui même ?